Compilerbau I
SS 2008

Leitfaden für die Prüfungsvorbereitung

Die folgenden Fragen sollen ein „Leitfaden“ zur Vorbereitung auf die Prüfung sein, das heisst diese Fragen sollte man sich selbst beim Lernen stellen, und folglich in der Prüfung beantworten können. Das bedeutet nicht, dass in der Prüfung nicht auch andere Fragen gestellt werden können, oder diese Fragen in abgewandelter Form. Es handelt sich nicht um einen vollständigen Fragenkatalog, den es gilt, auswendig zu lernen.

Kapitel I - Einführung

- Was versteht man allgemein unter einem Compiler? Wie unterscheiden sich Compiler und Interpreter? Wie sehen die Phasen des Kompilierens aus?

Kapitel II - Lexikalische Analyse

- Welchen Zweck erfüllt die lexikalische Analyse? (Stichworte: Lexem, Muster, Token)
- Wie sind reguläre Sprachen/Ausdrücke definiert? Man sollte mit regulären Ausdrücken umgehen können und einige kleinere Beispiele parat haben.
- Wie sehen die verwendeten Automatenmodelle aus?
- Wie übersetzt man einen regulären Ausdruck in einen endlichen Automat? Man sollte hier die beiden Verfahren beschreiben können.
- Minimalisierung eines DEA.
- Was ist die genaue Aufgabe des Scanners? Was unterscheidet ihn von einem reinen DEA?

Kapitel III - Syntaktische Analyse

Kontextfreie Grammatiken

- Wie sind kontextfreie Grammatiken definiert? Was versteht man unter den Begriffen Ableitung, Satzform, Links-/Rechtsableitung, Links-/Rechtssatzform, Ableitungsbaum? Was versteht man unter der Sprache einer Grammatik?
- Was versteht man unter ein-/mehrdeutigen Grammatiken? Man sollte an dieser Stelle auch Beispiele parat haben für ein-/mehrdeutige Grammatiken. Wie kann man Mehrdeutigkeit eliminieren? Ist dies immer möglich?
- Was versteht man unter (den beiden Arten von) Linksrekursion? Wie und warum eliminiert man sie?
- Was versteht man unter Linksfaktorisierung? Wozu braucht man sie?

Top-Down Parsing

- Was ist die Aufgabe eines Top-Down Parsers?
- Was versteht man unter Recursive Decent? Wie sieht ein nichtdet. Recursive Decent Parser aus?
- Wie löst man den Nichtdeterminismus auf? (Stichwort: Predictive Parsing) Wie sind *FIRST* und *FOLLOW* definiert? Wie berechnet man sie?
- Wann liegt eine Grammatik in $LL(1)$? Was bedeutet die Bezeichnung $LL(1)$? Wie sieht das Verfahren zur Konstruktion der Parsing-Tabelle aus? Wie erkennt man anhand der Parsing-Tabelle, ob die erzeugende Grammatik in $LL(1)$ liegt?
- Wie sieht der nichtdet. Kellerautomat für das Top-Down Parsing aus? Wie macht man ihn deterministisch?
- Welche Vor-/Nachteile besitzen Top-Down Parser? Man sollte vielleicht einige Beispiele parat haben um Antworten begründen zu können.

Bottom-Up Parsing

- Was ist die Aufgabe eines Bottom-Up Parsers?
- Was ist ein Reduktionsschritt? Was ist ein Handle? Welche Bedeutung hat ein Handle?
- Was ist ein viable prefix? Welche Bedeutung haben viable prefixes? Wie erkennt man viable prefixes?
- Was ist ein Item? Wann ist ein Item gültig für ein viable prefix?
- Was versteht man unter dem $LR(0)$-Automaten zu einer KFG? Was ist seine Funktion? Wie ist der Zusammenhang zu den viable prefixes und den gültigen Items?
- Wie wird der $LR(0)$-Automat benutzt um den Kellerautomaten zu steuern? Wie sehen die *GOTO*, *CLOSURE* und *Action* Funktionen aus im Falle eines SLR-Parser? Hier ist vorallem die *Action* Funktion wichtig, *GOTO* und *CLOSURE* sollte man aber intuitiv verstanden haben.
- Wann liegt eine Grammatik in SLR? Warum verwendet man in der Praxis nur selten SLR-Parser? Hier sollte man am besten Beispiele parat haben.
- Wie lässt sich die SLR-Methode verbessern? Wie sehen $LR(1)$-Items und der $LR(1)$-Parser aus? Wann liegt eine Grammatik in $LR(1)$? Gibt es auch Grammatiken, die nicht in $LR(1)$ liegen? Wiederum mit Beispielen belegen können.
- Welchen Nachteil haben $LR(1)$-Parser? Warum verwendet man sie nicht in der Praxis? Wie kann man diesen Nachteil umgehen? Wann liegt eine Grammatik in $LALR(1)$? Existieren Grammatiken, die in $LR(1)$ liegen, aber nicht in $LALR(1)$? Ebenfalls wieder Beispiele anbringen.
- Wie verwendet man mehrdeutige Grammatiken in Parsergeneratoren? Kann eine mehrdeutige Grammatik in SLR, $LR(1)$ oder $LALR(1)$ liegen? Wie löst man Shift-/Reduce-Konflikte mit Hilfe von Prioritäten bzw. Assoziativitäten? Wie werden die Prioritäten bzw. Assoziativitäten in der `yacc/JavaCUP`-Eingabe festgelegt? Was sind Pseudo-Tokens? Warum will man überhaupt mehrdeutige Grammatiken verwenden?